1. Consider a dataset consisting of 500 positive examples followed by 500 negative examples. As discussed in the book the perceptron algorithm will take a long time to converge (that is, it will ignore most of the positive examples before updating once a negative is observed). How does this observation not contradict the perceptron convergence bound?

The convergence bound gives an upper bound on the number of perceptron algorithm updates, not the number of iterations through the data set. Since the dataset is sorted in this way, convergence will take a long time due to the number of iterations, but the number of updates will not contradict the convergence bound.

2. We saw in class how the perceptron update rule moves the current estimate of the weight vector in a direction that maximizes signed distance of the current example to the classification hyperplane. However, there is no guarantee that the current example seen will be correctly classified. We can accomplish that by adding a step size \( \rho \) to the update rule. Given that example \( x_i \) is misclassified by current estimates \( w_{old} \) and \( b_{old} \), and an update rule of the form:

\[
w_{new} = w_{old} + \rho y_i x_i
\]

give a condition for \( \rho \) that ensures that \( x_i \) is classified correctly by \( w_{new} \) and \( b_{new} \).

For simplicity in this derivation, we will consider the unbiased perceptron. (The biased perceptron can be converted to the unbiased perceptron by adding an extra component with value 1 to \( x_i \) and an extra component to \( w \). The value of the extra component in \( w \) at each step is just the bias \( b \).

For a point that is misclassified, we have \( y_i (w_{old} \cdot x_i) < 0 \) and we want \( y_i (w_{new} \cdot x_i) > 0 \):
where we used the property that $y_i^2 = 1$ for binary classification with labels $+1$ and $-1$.

This condition on $\rho$ can easily be converted to the appropriate form for the biased perceptron:

$$\rho > -\frac{y_i(w_{old} \cdot x_i + b_{old})}{||x_i||^2 + 1}$$

3. Suppose a train a decision tree on dataset $D$. Now, I center and standardize all features in $D$ and train a decision tree again. Will the trees be different? Now consider the same question for the perceptron algorithm, will the two perceptron models be different?

The decision trees should be equivalent, although the split points will have different numerical values due to the transformation. The optimal split point for each feature will simply be transformed.

For the perceptron algorithm, the model may be different. If the data was originally linearly separable, it will still be linearly separable after the transformation, and the perceptron algorithm will converge on a separating hyperplane. However, the algorithm may update on different data points, and so the hyperplane may be different.

4. The book discusses using cross-validation to choose a hyper-parameter (instead of using a set-aside tuning set). Discuss the merits of using cross-validation to get an estimate of generalization error for a learning algorithm. For example, you divide the data into training/tuning/testing partitions 10 times at random and report on mean error rate over the ten testing partitions as your generalization error.

This will provide a better measure of generalization error. It’s possible that the original selection of the test dataset was biased in some way, and so the estimate of the generalization error will also be biased. Cross-validation done in this way will provide a better estimate of the generalization error. However, this comes at a computational cost, since testing now is carried out on 10 test partitions instead of just a single test partition.